Brewery Scheduling

After a warm summer day, Lea enjoys a nice cold beverage while sitting in the grass down by the river and watching the sunset. Like most of the people from the region she comes from, she usually enjoys a beer on these occasions. And after having a few sips of the exquisite golden liquid, she contemplates the work that is put behind brewing such a masterpiece. Thus, she decides to visit the **BIER** (**B**rewery of **I**nternational **E**xcellence and **R**elevance), one of the many local breweries, on the next day to learn a bit more about the process behind her favourite beverage. Apart from all the usual stuff about brewing, Lea learns about how the workers in the brewery are scheduled. There are many different tasks to be fulfilled, from unloading barley or hops to operate heavy machinery, and many workers that can fulfil these tasks. The brewery has clustered the tasks into several groups, where each group consists of a chain of tasks that has to be performed in succession, but is independent from tasks in other groups. Some of the tasks require the work of several workers simultaneously, but they all require one time unit. As quirky as Lea is, she thinks that the scheduling process and the schedule itself can be made better and after returning home, she immediately starts working on finding a perfect solution. And she might be right. For the sake of better beer, can you help Lea find an optimal personnel schedule?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with three integers n, m and d, with n being the number of tasks (indexed from 1 to n), m being the number of workers, and d being the deadline where all tasks need to be finished (i.e., the schedule starts at time 0 and should be finished upon timeslot d, i.e. at time d at the latest). n lines follow describing the tasks, where line i contains two integers c_i and p_i. c_i denotes the category of task i, i.e., how many workers are needed to perform task i, and p_i denotes the task that needs to be finished before the processing of task i can start. (A task with no predecessor task has $p_i = -1$.)

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is either: a schedule that takes at most d time units; or "impossible" if there is no such schedule. A schedule consists of a sequence $t_1\ t_2\ \ldots\ t_n$ where t_i is the timeslot task i is processed (e.g. timeslot j means that a task is started at time $j-1$ and finished its processing at time j) and at most m workers may be working simultaneously. If there is more than one solution, any of them will be accepted.

Constraints

- $1 \leq t \leq 50$
- $1 \leq n \leq 13$
- $1 \leq m \leq 50$
- $1 \leq d \leq 30$
- $1 \leq c_i \leq m$ for all $1 \leq i \leq n$
- $1 \leq p_i \leq n$ for all $1 \leq i \leq n$
- $1 \leq t_i \leq d$ for all $1 \leq i \leq n$

Sample Input 1

```
9
6 5 4
4 -1
3 1
1 2
3 -1
3 4
1 -1

5 6 6
3 -1
4 -1
4 -1
3 -1
3 -1

5 3 3
1 -1
3 1
2 2
3 -1
3 -1

3 1 6
1 -1
1 1
1 2

2 1 3
1 -1
1 1

1 1 4
1 -1

4 3 6
3 -1
1 -1
1 2
1 -1

3 8 5
5 -1
7 -1
3 -1

5 7 6
7 -1
7 1
6 2
7 -1
2 -1
```

Sample Output 1

```
Case #1: 1 2 3 3 4 1
Case #2: 1 3 4 1 2
Case #3: impossible
Case #4: 1 2 3
Case #5: 1 2
Case #6: 1
Case #7: 1 2 3 2
Case #8: 1 2 1
Case #9: 1 4 5 2 3
```